Programmieren in Java

Vorlesung 09: Generics and Comparison

Prof. Dr. Peter Thiemann

Albert-Ludwigs-Universität Freiburg, Germany

SS 2017

Inhalt

Lösungsbeispiel: w08-1 SearchTree

Zusatzinfo (vgl. Forum und Aufgabe)

- Ein binärer Suchbaum ist **entweder** ein Blatt **oder** ein innerer Knoten bestehend aus einem linken Teilbaum, einem Schlüssel s (hier: ein `int`) und einem rechten Teilbaum.
- Der linke Teilbaum ist ein Suchbaum, dessen Schlüssel alle kleiner als s sind.
- Der rechte Teilbaum ist ein Suchbaum, dessen Schlüssel alle größer als s sind.

Rekursive Klassen

Definition

Eine **rekursive Klasse** hat Felder, von denen aus Objekte der gleichen Klasse erreichbar sind.

Beispiel: Branch ist rekursive Klasse

```
1 public interface BTree {
2   public int size();
3 }
4 public class Branch implements BTree {
5   private final BTree left; // may hold another Branch object
6   private final BTree right; // may hold another Branch object
7   // ...
8 }
```

Demo Time

Aus der Praxis

Rational

```
1  public class Rational {
2      private long nominator;
3      private long denominator;
4      // ...
5      public Rational simplify() {
6          long a = this.nominator;
7          long b = this.denominator;
8          long c = 1;
9
10         while (b != 0) {
11             c = a % b;
12             a = b;
13             b = c;
14         }
15
16         this.nominator /= a;
17         this.denominator /= a;
18
19         return this;
20     }
21 }
```

Rational #2

```
1  public class Rational {
2      private long nominator;
3      private long denominator;
4      // ...
5      @Override
6      public boolean equals(Object r) {
7          if (r == this) {
8              return true;
9          }
10         if (!(r instanceof Rational)) {
11             return false;
12         }
13         Rational rational = (Rational) r;
14         return Objects.equals(this.toDouble(), rational.toDouble());
15
16     }
17 }
```

Generics und Vergleich

Ehe für alle

Aufgabe

Im Projekt “Ehe für alle” sollen beliebige Objekte heiraten können. Im Zuge der Gleichberechtigung gibt es zwei Rollen, `cris` und `cros`, die aber beliebig gewechselt werden können. Der Partner in der `cros` Rolle kann durch einen gleichartigen Partner ersetzt werden.

Ehe für alle

Aufgabe

Im Projekt "Ehe für alle" sollen beliebige Objekte heiraten können. Im Zuge der Gleichberechtigung gibt es zwei Rollen, `cris` und `cros`, die aber beliebig gewechselt werden können. Der Partner in der `cros` Rolle kann durch einen gleichartigen Partner ersetzt werden.

Modellierung

- Verwende eine Klasse `Couple` zur Modellierung des Ehepaars.
- Die Klasse besitzt zwei Felder, `cris` und `cros`.
- Die Methode `swapRoles` wechselt die Rollen.
- Die Methode `replace` ersetzt den `cros` Partner.

Modellierung

```
1  public class Couple<CRIS, CROS> {
2    private final CRIS cris;
3    private final CROS cros;
4    // constructor
5    public Couple<CROS, CRIS> swapRoles() {
6      ...
7    }
8    public Couple<CRIS, CROS> replace(CROS newCros) {
9      ...
10   }
11 }
```

- Die Klasse `Couple` ist **generisch**, zu erkennen an `<CRIS, CROS>`.
- `CRIS` und `CROS` sind zwei **Typvariablen**, die jeweils für einen beliebigen Referenztyp stehen.
- Die Typen der Methoden können auch Typvariablen enthalten.

Verwendung: Dog

```
1  public class Dog {
2     // predefined constructor: new Dog()
3  }
4  public class Cat {
5     // predefined constructor: new Cat()
6  }
7
8  ...
9  Couple<Cat, Dog> cd = new Couple<> (new Cat(), new Dog());
10 Couple<Dog, Cat> dc = cd.swapRoles()
11 Couple<Dog, Cat> ndc = dc.replace(new Cat());
12 ...
```

Spezialisierung

```
1 public class DogCouple extends Couple<Dog, Dog> { ... }
2 public class CatCouple extends Couple<Cat, Cat> { ... }
3 public class Human { ... }
4 public class HumanCouple extends Couple<Human, Human> {
5   public HumanCouple replace (Human newPartner) {
6     throw new IllegalOperation ("Couple first has to separate");
7   }
8 }
```

Vergleich

Aufgabe: Behalte den größten und kleinsten Wert

Minimax

Ein Minimax Thermometer misst regelmäßig die Temperatur und merkt sich dabei die minimale und maximale Temparatur.

Modellierung

- Eine Klasse `Minimax` mit zwei Feldern `minVal` und `maxVal`.
- Eine Methode `measured (int value)`, die ein eintreffende Messung verarbeitet.
- Eine Methode `reset (int value)`, die Minimum und Maximum auf `value` setzt.

Minimax

```
1  public class Minimax {
2    private int minVal;
3    private int maxVal;
4    public void measured(int value) {
5      if (value < minVal) { minVal = value; }
6      if (value > maxVal) { maxVal = value; }
7    }
8    public void reset (int value) {
9      minVal = value;
10     maxVal = value;
11   }
12 }
```

Aufgabe: Verallgemeinere Minimax auf beliebige Werte

Idee: Generics!

Generalisiere den Typ `int` zur Typvariable `V`.

```
1  public class Minimax<V> {
2    private V minVal;
3    private V maxVal;
4    public void measured(V value) {
5      if (value < minVal) { minVal = value; }
6      if (value > maxVal) { maxVal = value; }
7    }
8    public void reset (V value) {
9      minVal = value;
10     maxVal = value;
11   }
12 }
```

Aufgabe: Verallgemeinere Minimax auf beliebige Werte

Idee: Generics!

Generalisiere den Typ `int` zur Typvariable `V`.

```
1  public class Minimax<V> {
2    private V minVal;
3    private V maxVal;
4    public void measured(V value) {
5      if (value < minVal) { minVal = value; }
6      if (value > maxVal) { maxVal = value; }
7    }
8    public void reset (V value) {
9      minVal = value;
10     maxVal = value;
11   }
12 }
```

Ooops

Wird nicht akzeptiert, da < nur für Zahlen definiert ist!

Ausweg: Das vordefinierte Interface Comparable

```
1 package java.lang;
2 public interface Comparable<T> {
3   int compareTo (T specifiedObject);
4 }
```

Der Methodenaufruf `this.compareTo (specifiedObject)` ...

Compares this object with the specified object for order. Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Comparable

Verwendung

```
1        Integer i1 = new Integer (42);
2        Integer i2 = new Integer (4711);
3        int result = i1.compareTo (i2); // Zahlenvergleich
4        assertTrue (result < 0);
5
6        String s1 = "Affe";
7        String s2 = "Banane";
8        result = s1.compareTo (s2); // Stringvergleich, lexikographisch
9        assertTrue (result < 0);
```

Aufgabe, 2ter Versuch: Verallgemeinere Minimax

Idee: Generics!

Generalisiere den Typ `int` zur Typvariable `V`, **aber `V` muss mit anderen `V` vergleichbar sein**.

```
1  public class Minimax<V extends Comparable<V>> {
2    private V minVal;
3    private V maxVal;
4    public void measured(V value) {
5      if (value.compareTo (minVal) < 0) { minVal = value; }
6      if (value.compareTo (maxVal) > 0) { maxVal = value; }
7    }
8    public void reset (V value) {
9      minVal = value;
10     maxVal = value;
11   }
12 }
```

Aufgabe, 2ter Versuch: Verallgemeinere Minimax

Idee: Generics!

Generalisiere den Typ `int` zur Typvariable `V`, **aber `V` muss mit anderen `V` vergleichbar sein**.

```
1  public class Minimax<V extends Comparable<V>> {
2    private V minVal;
3    private V maxVal;
4    public void measured(V value) {
5      if (value.compareTo (minVal) < 0) { minVal = value; }
6      if (value.compareTo (maxVal) > 0) { maxVal = value; }
7    }
8    public void reset (V value) {
9      minVal = value;
10     maxVal = value;
11   }
12 }
```

Wird akzeptiert

Verwendung

```
1 Minimax<String> mm = new Minimax<>("M");
2 mm.measured ("Q");
3 mm.measured ("L");
4 mm.measured ("O");
5 mm.measured ("A");
6 assertEquals("A", mm.getMinVal());
7 assertEquals("Q", mm.getMaxVal());
```

Zusammenfassung

Zusammenfassung

- Generics zur Definition von Klassen, bei denen der Typ von manchen Feldern variieren kann.
- Generics können auch in Interfaces auftreten (vgl. Collection Framework)
- `Comparable` Interface zur Definition von Methoden und Klassen, bei denen die Typen generisch sind, aber eine Vergleichsoperation verwendet werden soll.
- Comparable ist oft eine Vorbedingung im Collection Framework.

Fragen